Online Meta-neuron based Learning Algorithm for a spiking neural classifier

Online Meta-neuron Based Learning Algorithm For A Spiking Neural Classifier

Shirin Dora[a], Sundaram Suresh[a,*], Narasimhan Sundararajan[a]

[a]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*

Abstract

This paper presents a new spiking neural network architecture with a meta-neuron which envelopes all the pre- and postsynaptic neurons in the network. The concept of the meta-neuron is inspired by the role of astrocytes in modulating synaptic plasticity in biological neural networks. The meta-neuron utilizes the global information stored in the network (synaptic weights) and the local information present in the input spike pattern to determine a weight sensitivity modulation factor for a given synapse. Based on the weight sensitivity modulation factor and the postsynaptic potential of a neuron, the meta-neuron based learning rule updates the synaptic weights in the network to produce precise shifts in the spike times of the postsynaptic neurons. Using this learning rule, an Online Meta-neuron based Learning Algorithm (OMLA) is presented for an evolving spiking neural classifier. The learning algorithm employs heuristic learning strategies for learning each input spike pattern. It can choose to add a neuron, update the network parameters or delete a spike pattern depending on the spike times of the output neurons. OMLA employs a meta-neuron with memory that stores only those spike patterns which are used to add a neuron to the network. These spike patterns (spike patterns in meta-neuron memory) are used as representative of past information stored in the network during subsequent neuron additions. The performance of OMLA has been compared with both the existing online learning and batch learning algorithms for spiking

*Corresponding author
Email address: ssundaram@ntu.edu.sg (Sundaram Suresh)

neural networks using the UCI machine learning benchmark data sets. The statistical comparison clearly indicates that the OMLA performs better than other existing online learning algorithms for spiking neural networks. Since, OMLA uses both, the global as well as the local information in the network, it is also able to perform better than other batch learning algorithms.

Keywords: Spiking neural networks, Evolving architecture, Meta-neuron, Pattern classification, Online Learning

---

1. Introduction

The neural circuitry in the brain is continuously updated to adapt the response of the neurons to future events. This property of the brain is termed as plasticity. It is the primary reason behind the ability of the brain to exhibit a wide range of functionalities and also for its eternal learning capabilities. These qualities have motivated the developments in the field of Spiking Neural Networks (SNNs) that try to emulate, both the behavioral and structural properties of the brain (biological neural networks).

SNNs are quite different from the earlier generations of artificial neural networks and as a result it is difficult to directly extend the existing neural network learning algorithms to SNNs. But, the ability of SNNs to mimic the feedforward sigmoidal neural networks [21] and their superior computational power in comparison to sigmoidal neurons [22] based networks have motivated researchers to develop new learning algorithms for SNNs. The mathematical models of spiking neural networks are closer to their biological counterparts, which encouraged researchers to look more deeply towards the learning phenomena observed in the brain.

Spike Timing Dependent Plasticity (STDP) [23, 7] is one of the most studied learning phenomenon observed in biological neural systems. It uses the temporal differences between the pre- and postsynaptic spike times for adapting the weights. This implies that STDP considers only the information that is locally available to a neuron for adapting the weights of the incoming synapses. In the

absence of any counterbalancing mechanism, a local learning rule like STDP would act as a positive feedback loop for potentiated synapses, thereby leading to their further potentiation. This can lead to the creation of regions in the brain, which contain highly active dendrites at the expense of the regions having inactive dendrites [16]. Hence, there is a need to develop a learning algorithm which can overcome this problem.

It is well-known that the synapses in our brain are capable of simultaneously exhibiting multiple forms of plasticity [6]. For example, it has been reported in the neuroscience area that astrocytes are star-shaped cells found in the brain that form an envelope around the synapses connecting the neurons in the brain. They can also make contacts with up to 100,000 synapses at a given time [5, 17], thereby allowing them to intercept and process the information transmitted across these synapses [1, 24]. This allows the astrocyte cells to utilize the global information in the network for modulating the sensitivities of multiple synapses simultaneously [29].

Inspired by the roles of astrocytes, in this paper, we develop a spiking neural network architecture with a newly defined concept of a meta-neuron. The meta-neuron envelops the pre- and the postsynaptic neurons. Also, there exists bidirectional communication between the meta-neuron and the enveloped synapses. Based on the global information contained in the network (synaptic weights) and the local information present in a synapse (presynaptic spikes), the meta-neuron modulates the sensitivities of all the synapses in the network. Each synapse changes its weight based on the weight sensitivity modulation factor generated by the meta-neuron and the required change in the postsynaptic potential such that the postsynaptic neuron spikes at the desired time. This is referred to as the meta-neuron based learning rule. It is a generic learning rule that updates the weights of the postsynaptic neurons in one-shot to produce precise shifts in their spike times.

Since the meta-neuron based learning rule can determine the appropriate adjustments in the synaptic weights of a postsynaptic neuron in one-shot, we propose an Online Meta-neuron based Learning Algorithm (OMLA) for pattern

classification problems that evolves the spiking neural network architecture and simultaneously adapts its synaptic weights. The learning algorithm employs three heuristic strategies to evolve the network and update the network parameters; they are the 'neuron addition strategy', the 'delete spike pattern strategy' and the 'parameter update strategy'. The appropriate learning strategy for a given spike pattern is selected based on the first spike generated by the output layer neurons. In case of neuron addition strategy, a new neuron is added to the network when the knowledge stored in the network is not sufficient to approximate the information present in the current input spike pattern. The weights of the newly added neuron are initialized to the normalized contributions of the corresponding input neurons for the current input spike pattern. Based on the initialized weights, the threshold of the new neuron is set as its postsynaptic potential at the target firing time. The spike patterns used by the learning algorithm to evolve the network (add a neuron) are also stored in meta-neuron memory. The learning algorithm uses these spike patterns as pseudo-inputs to efficiently capture the past knowledge while adding a neuron for a subsequent spike pattern. For the delete spike pattern strategy, the learning algorithm discards a spike pattern from the learning process when it is similar to the previously learned spike patterns. In the parameter update strategy, the synaptic weights are adapted using the meta-neuron based learning rule.

To illustrate the effect of the 'delete spike pattern strategy' and the meta-neuron memory on the performance of OMLA, a study is conducted using the Ionosphere problem from the UCI machine learning repository [20]. As highlighted in a previous work [32], the study showed that deleting similar spike patterns improves the generalization performance of the learning algorithm. The study also showed that the performance of the learning algorithm is significantly better when meta-neuron memory is used for approximating the past knowledge. The Ionosphere problem is also used to analyze the impact of its algorithm parameters on the performance of the learning algorithm. Based on this study, guidelines have been suggested for setting the algorithm parameters to appropriate values.

A detailed comparison on multiple benchmark problems has also been done between the performance of OMLA and other existing online learning algorithms for SNNs, namely, online spiking neural network [35] and the online version of Self-Regulating Evolving Spiking Neural (SRESN) classifier [8]. The performance of OMLA has been statistically compared with the performance of other online learning algorithms using one-way ANOVA [13] test followed by a pairwise comparison using the Bonferroni test [11, 12]. The results of performance comparison indicate that OMLA performs better than the other online learning algorithms in a 95% confidence interval. For completeness, the performance of the OMLA is also compared with three well-known batch learning algorithms for SNNs, viz. SpikeProp [4] Synaptic Weight Association training (SWAT) [34] and the batch version of SRESN classifier. The performance of OMLA is better in comparison with batch learning algorithms as well as it utilizes both the local as well as global information in the network.

Rest of the paper is organized as follows. Section 2 provides a brief overview of the existing learning algorithms in the spiking neural network literature. Section 3 describes the newly introduced spiking neural network with a meta-neuron. Section 4 presents the online meta-neuron based learning algorithm. Section 5 describes the working of the learning algorithm followed by the results of a detailed performance evaluation of the learning algorithm for multiple benchmark classification problems. Section 6 summarizes the conclusions from this study.

2. Related Works

Broadly, the existing spiking neural network learning algorithms for pattern classification problems can be classified into three major categories, namely, the gradient-descent based learning algorithms [4, 14], the rank order based learning algorithms [19, 10, 37, 8, 18] and the Spike Timing Dependent Plasticity (STDP) [23, 7] based algorithms [25, 30, 35, 34].

One of the earliest spiking neural network learning algorithm was the gra-

dient based learning approach, called SpikeProp [4]. It was an extension of the traditional error back-propagation learning rule of feedforward networks to spiking neural networks. Chronotron [14] is another gradient based learning algorithm which estimates the synaptic weights by minimizing the distance between the actual and desired output spike trains. SpikeProp and Chronotron encode information using the precise time of the spikes. Rank order coding [28] is another coding strategy that encodes information using the order of the spikes. Due to the ability of rank order coding for faster information transmission [33] several learning algorithms have been developed for SNNs using rank order coding [19, 9, 10]. It is a non-local encoding technique, but it allows the network to learn the knowledge present in the spike patterns in one-shot. On the other hand, spike timing dependent plasticity is a local learning technique considered to be the phenomenon behind learning in biological neural systems. It has been the basis of many learning algorithms for SNNs [25, 35, 34].

Remote Supervised Method [25] employs a combination of Spike Timing Dependent Plasticity (STDP) and anti-spike timing dependent plasticity for updating the weights in a spiking neural network. In [35], STDP along with an unsupervised Hebbian like learning rule was used to train a network with a single evolving hidden layer in an online framework. Similar to other Hebbian learning mechanisms, STDP is also inherently unstable due to its local nature [26, 15]. It can form a positive feedback loop between a neuron and its dendrites leading to further potentiation of potentiated synapses [16]. This may result in an unbalanced distribution of weights. Synaptic Weight Association Training (SWAT) [34] combined STDP with Bienenstock Cooper Munro [3] learning rule to overcome the unstable nature of STDP. But, the update rule for SWAT does not utilize the global information stored in the network. As a result, SWAT may not produce precise shifts in the spike times of the output neurons.

In the next section, a new spiking neural network architecture and its learning rule are presented that can precisely shift the spike times of the postsynaptic neurons by performing a one-shot update of the synaptic weights in the network.

3. Spiking Neural Network with a Meta-neuron

In this section, the architecture and the learning rule for a Spiking Neural Network (SNN) with a meta-neuron are described. The concept of the meta-neuron is inspired by the heterosynaptic plasticity induced by astrocyte cells in biological systems. Astrocyte cells can be simultaneously connected to multiple synapses [5, 17] which allows them to intercept the activities on the connected synapses and modulate their plasticity [1, 24]. This form of heterosynaptic plasticity demonstrated by astrocytes provides a good mechanism to consider the global information present in the network while updating synaptic weights.

The ideas presented in this section are applicable to a network with multiple postsynaptic neurons. But, for better understanding, let us consider a SNN that consists of m presynaptic neurons connected to a single postsynaptic neuron. The presynaptic neurons in the network are also connected to a single meta-neuron which allows the meta-neuron to access the local information present in the input spike trains. Further, the meta-neuron can access the global information stored in the network as synaptic weights of the postsynaptic neuron. Figure 1 shows the architecture of a spiking neural network with a meta-neuron.

The SNN is presented with m-dimensional spike patterns represented by $\mathbf{x} = \{x_1, \cdots, x_i, \cdots, x_m\}$ at intervals of time T, where T is termed as the simulation interval for one spike pattern. Here, $x_i = \{t_i^{(1)}, \cdots, t_i^{(g)}, \cdots, t_i^{(G_i)}\}$ is the spike train, with G_i spikes, generated by the i^{th} presynaptic neuron. The unweighted Postsynaptic Potential (PSP) induced by the g^{th} spike generated by the i^{th} presynaptic neuron at time t is given by $\epsilon(t - t_i^{(g)})$. In this paper, $\epsilon(.)$ has been modelled using the spike response function [4], given by

$$\epsilon(s) = \frac{s}{\tau} \exp\left(1 - \frac{s}{\tau}\right) \tag{1}$$

where τ is the time constant for the neuron and is set to 3 ms. Based on the PSPs induced by the individual presynaptic neurons, the PSP (v) of the postsynaptic neuron at time t is given by

$$v(t) = \sum_i \sum_g w_i \epsilon(t - t_i^{(g)}) \tag{2}$$

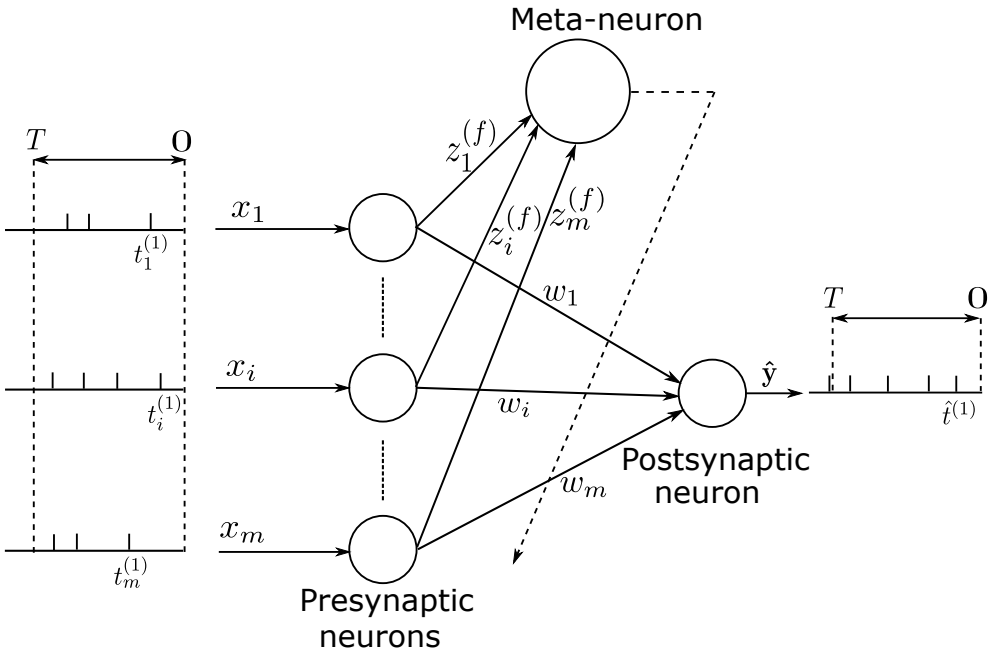


Figure 1: Architecture of a two layered spiking neural network with a meta-neuron. The SNN has m presynaptic neurons connected to a single postsynaptic neuron. The presynaptic neurons are also connected to the meta-neuron, which allows the meta-neuron to access the local information present in the input spike train on a particular synapse. The meta-neuron can also access the global information (synaptic weights) stored in the network, represented by the oblique arrow from the meta-neuron across the network.

where w_i is the weight of the synapse between the i^{th} presynaptic neuron and the postsynaptic neuron. The postsynaptic neuron generates a spike whenever its PSP reaches a threshold θ and immediately after generating a spike, the PSP of the postsynaptic neuron is reset to zero. The output of the postsynaptic neuron is a spike train $\hat{y} = \{\hat{t}^{(1)}, \cdots, \hat{t}^{(f)}, \cdots\}$. The objective of the learning rule is to closely capture the functional relationship between the input spike patterns and the desired output spike patterns. For this purpose, the weights of the postsynaptic neuron are updated for each spike generated by the postsynaptic neuron individually. Suppose $t^{(f)}$ represents the desired time of the f^{th} spike generated by the postsynaptic neuron then the learning rule performs a one-shot update of the weights of the postsynaptic neuron such that it generates a spike

precisely at $t^{(f)}$.[1]

To ensure that the postsynaptic neuron generates a spike at $t^{(f)}$ for the current spike pattern ($\mathbf{x}$), its weights should be updated such that its PSP is equal to θ at $t^{(f)}$. Thus, the required change in PSP ($\Delta v^{(f)}$) of the postsynaptic neuron is given by

$$\Delta v^{(f)} = \theta - v(t^{(f)}) \tag{3}$$

Since only the presynaptic spikes in the interval $\Gamma = [\hat{t}^{(f-1)}, t^{(f)}]$ will contribute to a postsynaptic spike at $t^{(f)}$, the weights of the postsynaptic neuron are updated such that

$$\sum_i \sum_{t_i^{(g)} \in \Gamma} \Delta w_i \epsilon(t^{(f)} - t_i^{(g)}) = \Delta v^{(f)} \tag{4}$$

where Δw_i is the change in the synaptic weight of the connection between the i^{th} presynaptic neuron and the postsynaptic neuron. The weight update using Equation (4) ensures that the postsynaptic neuron generates a spike precisely at $t^{(f)}$.

In a given learning step (for a particular postsynaptic spike), the value of Δw_i depends on the contribution of the particular presynaptic neuron towards the required change in PSP ($\Delta v^{(f)}$). For this purpose, the meta-neuron estimates a weight sensitivity modulation factor for the synapses in the network which is used to determine the proportion of $\Delta v^{(f)}$ that is contributed by a given presynaptic neuron. It uses both local and global information to compute its weights ($\mathbf{z}^{(f)} = [z_1^{(f)}, \cdots, z_i^{(f)}, \cdots, z_m^{(f)}]$) and determine the weight sensitivity modulation factor ($M_i^{(f)}$) for each synapse of the postsynaptic neuron.

Based on the required change in PSP (Equation (3)) and the weight sensitivity modulation factor ($M_i^{(f)}$) of a synapse, the weights of the postsynaptic

[1]It is possible that the number of desired spikes (F) and the number of actual spikes ($\hat{F}$) generated by the postsynaptic neuron are not equal. In this case, the following convention is used to update the synaptic weights of the postsynaptic neuron. When $\hat{F} < F$, the time of the missing actual spikes is set to $(T + \delta)$ and in case $\hat{F} > F$, the desired time of extra spikes generated by the postsynaptic neuron is set to $(T + \delta)$. Here, $\delta > 0$ is a small positive number and a spike time of $(T + \delta)$ implies an absence of spike in the simulation interval.

neuron are updated using the *meta-neuron based learning rule* which is given by

$$\Delta w_i = M_i^{(f)} \frac{\Delta v^{(f)}}{\sum_{t_i^{(g)} \in \Gamma} \epsilon(t^{(f)} - t_i^{(g)})}, \qquad i \in \{1, \cdots, m\} \tag{5}$$

Here, $M_i^{(f)}$ is computed by the meta-neuron as the ratio of the PSP induced by the i^{th} presynaptic neuron at $t^{(f)}$ and the total PSP of the meta-neuron at $t^{(f)}$, i.e.,

$$M_i^{(f)} = \frac{\sum_{t_i^{(g)} \in \Gamma} z_i^{(f)} \epsilon(t^{(f)} - t_i^{(g)})}{\sum_i \sum_{t_i^{(g)} \in \Gamma} z_i^{(f)} \epsilon(t^{(f)} - t_i^{(g)})}, \qquad i \in \{1, \cdots, m\} \tag{6}$$

where $z_i^{(f)}$ is the weight of the synapse between the i^{th} presynaptic neuron and the meta-neuron. The meta-neuron sets its weights by comparing the information present in a given input spike train with the knowledge stored in the corresponding synaptic weight. In a given learning step, the synaptic weight of the connection between the i^{th} presynaptic neuron and the meta-neuron is set as

$$z_i^{(f)} = \begin{cases} u_i^{(f)}(t^{(f)}) - w_i & \text{if } u_i^{(f)}(t^{(f)}) > w_i \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, \cdots, m \tag{7}$$

Here, $u_i^{(f)}$ represents the knowledge present in the input spike train generated by the i^{th} presynaptic neuron. It is termed as the normalized PSP induced by the i^{th} presynaptic neuron at $t^{(f)}$ and is given by

$$u_i^{(f)}(t^{(f)}) = \frac{\sum_{t_i^{(g)} \in \Gamma} \epsilon(t^{(f)} - t_i^{(g)})}{\sum_i \sum_{t_i^{(g)} \in \Gamma} \epsilon(t^{(f)} - t_i^{(g)})} \tag{8}$$

It can be observed from Equation (7) that the meta-neuron weights are initialized to zero for a synapse when the normalized PSP of the corresponding input neuron for the current spike pattern ($\mathbf{x}$) at ($t^{(f)}$) is lower than its existing weight. The input neurons whose weights are initialized to zero will have a weight sensitivity modulation factor of zero and, hence, their weights are not updated. The selective update mechanism of the meta-neuron based learning rule is similar to the selective regulation of synaptic plasticity exhibited by the

astrocytes in the brain. This selective approach to modulating synaptic plasticity prevents a continuous increase in the weights of the potentiated synapses that is induced by STDP [16].

The intuitive motivation for computing $M_i^{(f)}$ using Equation (6) is to produce a higher change in the weight of the synapse when higher PSP is contributed by the corresponding presynaptic neuron. Further, the computation of $M_i^{(f)}$ using Equation (6) ensures that $M_i^{(f)}$ always lies in the interval $[0, 1]$ and $\sum_i M_i^{(f)} = 1$. This guarantees that the weight update results in a change of $\Delta v^{(f)}$ in the PSP of the postsynaptic neuron at $t^{(f)}$, thereby ensuring that the postsynaptic neuron generates a spike precisely at $t^{(f)}$.

The meta-neuron based learning rule (Equation (5)) is a generic learning rule that utilizes the local and global information in the network to perform a one-shot weight update of the synapses in the network such that the spike times of the postsynaptic neuron are precisely shifted. This property of the meta-neuron based learning rule makes it suitable for online learning. Further, determining the minimal structure required to approximate the relationship between the input and output spike patterns is a challenge in a spiking neural network. Hence, in the next section, we propose an online meta-neuron based learning algorithm for an evolving spiking neural classifier that employs the meta-neuron based learning rule to update the synaptic weights in the network.

4. Online Meta-neuron based Learning Algorithm for Pattern Classification Problems

In this section, an Online Meta-neuron based Learning Algorithm (OMLA) for a two layered evolving spiking neural network for pattern classification problems is developed. OMLA learns from input spike patterns in an online manner i.e. training spike patterns are presented to the network one-by-one and only once. The aim of the learning algorithm is to closely approximate the relationship between the input spike patterns and the corresponding class labels and evolve the network structure automatically. The predicted class for a given

spike pattern is determined based on the class association of that output neuron which spikes first (output neuron having a minimum latency). As a result, there is a need to encode the true class label in the form of a spike pattern. Since latency of the neuron is used to determine the class label, the learning algorithm trains the network to generate the first spike for a given spike pattern at the target firing time. The target firing time for a neuron belonging to the true class is set to a fixed time instant in the simulation interval, denoted by T_{ID}. For other class neurons, the target firing time is set to $T + \delta$, which implies that other class neurons should not generate a spike within the simulation interval.

OMLA utilizes the meta-neuron based learning rule as given in Equation (5) to learn and evolve the network in an online framework. It chooses one of the three different heuristic strategies, namely, 'neuron addition strategy', 'delete spike pattern strategy' or 'parameter update strategy' for learning a given input spike pattern. The suitable learning strategy is chosen based on the current input spike pattern and the knowledge present in the network. A similar self-regulated learning approach has been previously used in the Self-Regulating Evolving Spiking Neural (SRESN) classifier [8] which uses rank order learning to update the synaptic weights in a batch learning environment. Different from the SRESN classifier, OMLA learns from each spike pattern in one-shot using the meta-neuron based learning rule.

For an efficient approximation of the functional relationship between the input spike patterns and their class labels, the input spike patterns have to be presented to the network multiple times. To overcome this problem in online learning, OMLA employs a meta-neuron with memory that stores the spike patterns used to add new neurons to the network. While adding a neuron for a subsequent spike pattern, these pseudo-inputs (spike patterns in meta-neuron memory) are used by the learning algorithm for a better approximation of the past knowledge stored in the network.

Without loss of generality, it is assumed that the network has K output neurons, which were added while learning the spike patterns $\mathbf{x}_1, \cdots, \mathbf{x}_h, \cdots, \mathbf{x}_K$. At this juncture, the meta-neuron's memory will contain these spike patterns

that have been used to add the K neurons. Next, the current spike pattern $\mathbf{x}$ from class c is presented to the network for learning.

In the following discussion, CC (CC implies correct class) is used to represent the output neuron having minimum latency from class c. Suppose c_j is the class associated with the j^{th} output neuron, then the neuron CC is given by

$$CC = \underset{j,c_j=c}{\operatorname{argmin}} \hat{t}_j^{(1)} \tag{9}$$

Similarly, MC is used to represent the output neuron with minimum latency from any other class, given as

$$MC = \underset{j,c_j\neq c}{\operatorname{argmin}} \hat{t}_j^{(1)} \tag{10}$$

Since the learning algorithm uses only the first spike for learning and prediction, the discussion below uses $\hat{t}_j$ (instead of $\hat{t}_j^{(1)}$) and t_j (instead of $t_j^{(1)}$) to represent the actual and desired times of first spike generated by the j^{th} output neuron. Further, the normalized PSP induced by the i^{th} presynaptic neuron is denoted by u_i (instead of $u_i^{(1)}$). Next, the different learning strategies employed by the learning algorithm are given in detail.

- **Neuron addition strategy**: In this strategy, a new neuron is added to the network when the current spike pattern contains a significant amount of new knowledge. For this purpose, the learning algorithm considers the time interval between T_{ID} and $\hat{t}_{CC}$. A high value of this time interval ($T_{ID} << \hat{t}_{CC}$) implies that the knowledge stored in the network is not sufficient to approximate the current input spike pattern and hence, a new neuron is added to the network. A fixed time instant $T_n \in [T_{ID}, T]$ is used as a threshold for $\hat{t}_{CC}$ to develop a heuristic criterion for this strategy, given as

  If $\hat{t}_{CC} > T_n$

  Then a neuron is added to the network (11)

  where T_n is given by

  $$T_n = \alpha_n T + (1 - \alpha_n) T_{ID} \tag{12}$$

Here, α_n is termed as novelty threshold and is always set to a value in the interval $[0, 1]$. If its value is set closer to zero, the learning algorithm adds a neuron to the network for each input spike pattern resulting in overfitting. If it is set closer to one, the learning algorithm adds too few neurons to the network, resulting in an imprecise model of the data. It should be set to a value closer to one, to ensure proper generalization of the trained network on unseen spike patterns. A suitable range for initializing α_n is $[0.7, 1]$.

The weight of the synapse between the i^{th} input neuron and the newly added neuron is initialized according to the normalized PSP (Equation (8)) induced by the i^{th} input neuron at T_{ID}. Hence, the weights ($\mathbf{w}_{(K+1)} = [w_{1(K+1)}, \cdots, w_{i(K+1)}, \cdots, w_{m(K+1)}]$) of a newly added neuron are given as

$$w_{i(K+1)} = u_i(T_{ID}) \tag{13}$$

The threshold ($\theta_{(K+1)}$) for the neuron is initialized as

$$\theta_{(K+1)} = \sum_i \sum_g w_{i(K+1)} \epsilon(T_{ID} - t_i^{(g)}) \tag{14}$$

The learning algorithm considers the spike patterns used to evolve the network (spike patterns stored in meta-neuron memory) as pseudo-inputs representing the past knowledge stored in the network. These pseudo-inputs are used to update the weights of the newly added neuron, such that it closely approximates the past knowledge stored in the network.

Suppose $\hat{t}_{K+1}^{[h]}$ is the time of the first spike generated by the $(K+1)^{th}$ neuron for the h^{th} spike pattern in the meta-neuron memory. When $\hat{t}_{K+1}^{[h]}$ is closer to $\hat{t}_h^{[h]}$, the weights of the newly added neuron are updated such that it fires late for spike patterns from the class c_h. For this purpose, a fixed time duration, T_m is used to develop a criterion for updating the

weights of the newly added neuron given as

$$\begin{aligned} &\textbf{If } (\hat{t}^{[h]}_{K+1} - \hat{t}^{[h]}_{h}) < T_m, \qquad h \in \{1, \cdots, K\}, c_h \neq c_{K+1} \\ &\textbf{Then } \text{update the weights of the } (K+1)^{th} \text{ neuron} \end{aligned} \tag{15}$$

where T_m is given by

$$T_m = \alpha_m (T - T_{ID}) \tag{16}$$

Here, α_m is termed as the margin threshold and is always initialized to a value in the interval $[0, 1]$. If it is set closer to zero, the network will not generalize well due to smaller interclass margin. If it is set closer to one, then the network may not accurately approximate the knowledge acquired from past spike patterns very well. The experimental analyses showed that the performance of the learning algorithm is acceptable when α_m is set in the interval $[0, 0.3]$. The value of α_m is set to 0.3 for all simulations described in this work.

The weights of the newly added neuron are updated using the meta-neuron based learning rule and its desired spike time ($t^{[h]}_{K+1}$) for the h^{th} spike pattern in meta-neuron memory is given as

$$t^{[h]}_{K+1} = \hat{t}^{[h]}_{h} + T_m \tag{17}$$

- **Delete spike pattern strategy**: In this strategy, the learning algorithm deletes a spike pattern when a neuron from the same class as that of the current input spike pattern fires closer to the target firing time (T_{ID}), which implies that this particular spike pattern is similar to the earlier learnt spike patterns. This helps OMLA in avoiding over-fitting and generalizing better on unseen spike patterns. The learning algorithm uses a fixed time instant $T_d \in [T_{ID}, T]$ to develop a heuristic criterion for this strategy, given as

$$\begin{aligned} &\textbf{If } \hat{t}_{CC} \leq T_d \quad \& \quad (\hat{t}_{MC} - \hat{t}_{CC}) \geq T_m \\ &\textbf{Then } \text{the current input spike pattern is deleted} \end{aligned} \tag{18}$$

where T_d is given as

$$T_d = \alpha_d T + (1 - \alpha_d) T_{ID} \tag{19}$$

Here, α_d is termed as the delete threshold and is used to determine the spike patterns that can be discarded from the learning process. It is always set to a value in the interval $[0, 1]$. If it is set closer to zero, it will result in all spike patterns being learnt by the learning algorithm which will lead to a lower generalization performance. If it is set closer to one, it will result in the deletion of too many spike patterns resulting in an imprecise model of data. Based on the simulation studies, it was observed that a suitable range for α_d is $[0, 0.25]$. The performance of the learning algorithm is satisfactory when α_d is set in this interval. Its values is fixed at 0.25 for all the simulations presented in this work.

- **Parameter update strategy**: The learning algorithm chooses to update the synaptic weights of existing neurons when the criterion for none of the above strategies of neuron addition or delete spike pattern are satisfied. The aim of this strategy is to update the synaptic weights such that $\hat{t}_{CC}$ is closer to T_{ID} and there exists a high time difference between $\hat{t}_{CC}$ and $\hat{t}_{MC}$, for all the spike patterns. The weights of the neuron CC are updated to ensure that the correct class neuron fires closer to T_{ID}. Further, the learning algorithm also updates the weights of the neuron MC to ensure a higher time difference exists between $\hat{t}_{CC}$ and $\hat{t}_{MC}$.

  The weights of the neuron CC are updated using the meta-neuron based learning rule only when $\hat{t}_{CC}$ is higher than T_d. In this case, the desired spike time (t_{CC}) of the neuron CC for the current input spike pattern ($\mathbf{x}$) after the weight update is given as

$$t_{CC} = \hat{t}_{CC} - \alpha_s \hat{t}_{CC} \tag{20}$$

  where α_s is termed as the learning rate and is always initialized to a value in the interval $[0, 1]$. A high value of the learning rate causes oscillations in the learning process. Hence, a suitable range for initializing α_s is $[0, 0.1]$.

Algorithm 1 Pseudocode for Online Meta-neuron based Learning Algorithm

for each training spike pattern **do**
  $\hat{t}_{CC} \leftarrow$ spike time of same class neuron with minimum latency
  $\hat{t}_{MC} \leftarrow$ spike time of differnt class neuron with minimum latency
  if $\hat{t}_{CC} > T_n$ **then**
    Add a neuron
  else if $(\hat{t}_{CC} \leq T_d)$ & $(\hat{t}_{MC} - \hat{t}_{CC}) \geq T_m$ **then**
    Delete the spike pattern
  else
    if $\hat{t}_{CC} > T_d$ **then**
      Update the CC neuron
    end if
    if $(\hat{t}_{MC} - t_{CC}) < T_m$ **then**
      Update the MC neuron
    end if
  end if
end for

To improve the margin between $\hat{t}_{CC}$ and $\hat{t}_{MC}$, the weights of the neuron MC are updated using the meta-neuron based learning rule when $(\hat{t}_{MC} - t_{CC}) < T_m$. The desired spike time (t_{MC}) for the neuron MC after the weight update is given by

$$t_{MC} = t_{CC} + T_m \tag{21}$$

The utilization of global as well as local information by the meta-neuron enables the learning algorithm to estimate the changes in weights such that the relationship between the input spike patterns and the corresponding class labels is closely approximated in one-shot. Further, a meta-neuron with memory for storing spike patterns used to evolve the network allows the learning algorithm to approximate the past knowledge properly while adding a neuron.

A summary of the online meta-neuron based learning algorithm in a pseu-

docode format is given in the Algorithm 1. Next, the performance of the learning algorithm is evaluated on benchmark problems from the UCI machine learning repository and the results of evaluation are compared with that of other spiking neural classifiers.

5. Performance Evaluation of the Online Meta-neuron based Learning Algorithm

In this section, the performance of the learning algorithm is evaluated using five benchmark data sets from the UCI machine learning repository [20]. For all the simulation studies reported in this section, the spiking neurons in the output layer are modeled using the spike response model [4] and the time constant for the neuron is fixed at 3 ms. The real valued data from the benchmark problems is encoded into spike patterns using the population coding scheme [4]. As described in [8], the overlap constant for the population coding has been fixed at 0.7 and six receptive fields have been used for converting the real valued data into spike patterns. Using the population coding, each receptive field generates a spike in the interval $[0, 3]$ ms. Hence, the simulation interval has been set slightly higher than the range of input spikes, at 3.2 ms, to ensure that all spikes generated by the output layer neurons are recorded.

The performance of all the algorithms is evaluated based on overall training and testing accuracy, which is equal to the percentage of total number of spike patterns that are correctly classified by the network. The average performance over ten random trials is used for the purpose of comparison. All the experiments have been carried out in Windows 7 in MATLAB 2014b using a CPU with 12 logical cores, 16 GB memory with a speed of 3.2 GHz. Before discussing the results of the detailed performance evaluation, the working of the learning algorithm is described using the Ionosphere problem from the UCI machine learning repository. The Ionosphere problem is also used to describe the effect of the different algorithm parameters on the performance of the learning algorithm, based on which suitable guidelines are suggested for setting the parameters to

appropriate values.

5.1. Ionosphere Problem

The Ionosphere problem contains radar information collected from 16 high frequency receivers. The data set has in total 34 attributes and the problem is to determine whether the received signal conveys any information about the structure of the Ionosphere. It has a total of 351 spike patterns, out of which 175 are used for training and the rest for testing.

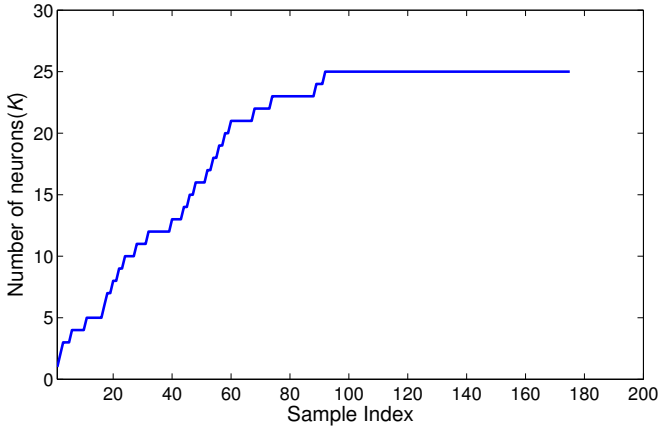


Figure 2: Neuron growth history of the learning algorithm for the Ionosphere problem

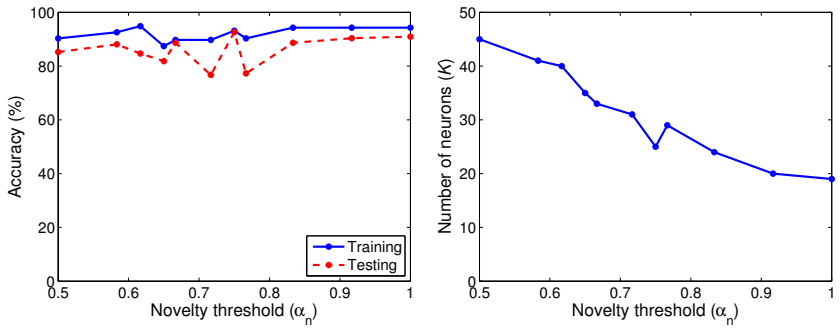
The learning algorithm has four main parameters, namely, a novelty threshold (α_n), a margin threshold (α_m), a delete threshold (α_d) and the learning rate (α_s). As described earlier, the two parameters, viz. the margin threshold and the delete threshold values are fixed at 0.3 and 0.25 respectively. For initializing the novelty threshold and learning rate, the suitable ranges have been indicated earlier as $[0.7, 1]$ and $[0, 0.1]$ respectively. For example, when the novelty threshold and the learning rate are initialized to 0.73 and 0.09 respectively, the average training accuracy is 93.2% and the average testing accuracy is 93% for the Ionosphere problem. Out of the 175 spike patterns, the learning algorithm used only 136 spike patterns for learning and added 25 neurons to the network. Figure 2 shows the neuron growth history for the Ionosphere problem.

The learning algorithm adds the last neuron for the 92^{nd} training spike pattern and chooses only to update the network parameters or delete the spike pattern from the learning process for the subsequent training spike patterns. Next, the effects of the 'delete spike pattern strategy' and the meta-neuron memory on the learning algorithm are described. Also, the impact of the algorithm parameters on the performance of the learning algorithm is illustrated.

Effects of the delete spike pattern strategy: Similar to the observed behavior in [32], it was observed that the learning algorithm achieves higher generalization accuracy when similar spike patterns are deleted. For the Ionosphere problem, training and testing accuracy of 93.2% and 93% are obtained when the learning algorithm is trained with the 'delete spike pattern strategy'. The learning algorithm deleted 39 spike patterns from the training set of 175 spike patterns (22% is deleted). When the learning algorithm was trained without the 'delete spike pattern strategy' all the spike patterns were used in the training. In this case, a training and testing accuracy of 93.8% and 84.6%, respectively are obtained. This clearly shows that the 'delete spike pattern strategy' helps in improving the generalization performance of the learning algorithm.

Effect of meta-neuron memory: A similar study was conducted to analyze the impact of meta-neuron memory on the performance of the learning algorithm. When the learning algorithm is trained without the meta-neuron memory, the average training and testing performance are 85.71% and 79.55%, respectively, while with the memory they are 93.2% and 93.0% respectively. This clearly highlights that, when the learning algorithm is trained without meta-neuron memory its performance drops considerably for the Ionosphere problem. In the absence of meta-neuron memory, the learning algorithm has no information about past knowledge stored in the network. As a result, newly added neurons do not approximate the past knowledge effectively. This results in lower performance in an online framework. The results clearly show that meta-neuron memory plays a vital role in improving the performance of the learning algorithm.

Effect of novelty threshold: To illustrate the effect of the novelty thresh-

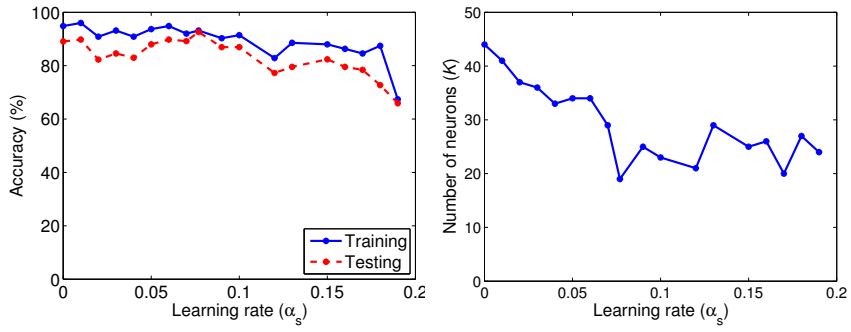


(a) (b)

Figure 3: Effect of novlety threshold (α_n) on the (a) training and testing accuracy, (b) number of neurons required to approximate the decision function

old (α_n) on the learning algorithm, it was evaluated for values of α_n in the range $[0.5, 1]$. Figure 3a shows both training and testing performance against α_n. It can be seen from the figure that there is a small change in the training performance whereas there is a change of over 15% in testing performance as α_n is varied over the interval $[0.5, 1]$. Figure 3b shows the impact of α_n on the number of neurons added by the learning algorithm to the network. It is seen that a lower value of α_n results in more neurons being added to the network and vice-versa, thereby, showing that α_n significantly impacts the generalization performance and the number of neurons added by the learning algorithm. Hence, it has to be chosen carefully. A suitable range for setting α_n is $[0.7, 1]$ to achieve good performance using a compact network.

Effect of learning rate: In this experiment, the choice of learning rate (α_s) in the interval $[0, 0.2]$ is studied. Figure 4a shows the impact of learning rate on both, the training and testing performance. It is seen that there is a small variation in the training as well as testing performance when the learning rate is in the range $[0, 0.1]$. When α_s is increased beyond 0.1 both the training as well as testing performance start deteriorating. This is because for a high value of α_s the network looses knowledge gained from previous spike patterns. Figure 4b shows the variation in the number of neurons added by the learning



(a) (b)

Figure 4: Effect of learning rate (α_s) on the (a) training and testing accuracy, (b) number of neurons required to approximate the decision function

algorithm as α_s is varied in the range $[0, 0.2]$. When α_s is set to zero, it plays no role in the learning process. In such a scenario, more neurons are required by the network to ensure that the knowledge present in the spike patterns is properly learned by the network. This is evident from the plot shown in Figure 4b. To summarize, α_s affects both, the generalization performance as well as the architecture of the trained neural network. Based on these observations, it is recommended that α_s be set in the range $[0, 0.1]$ for achieving good network performance.

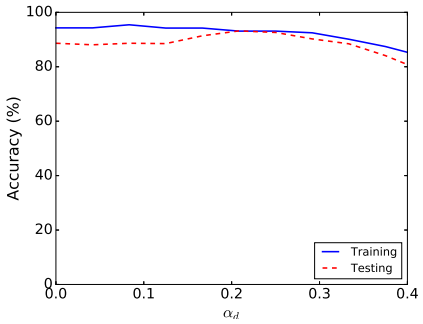


Figure 5: Effect of delete threshold (α_d) on the training and testing accuracy of OMLA

Effect of deletion threshold (α_d): In this experiment, the performance

of the Online Meta-neuron based Learning Algorithm (OMLA) is evaluated for multiple values of α_d in the interval $[0, 0.4]$. Figure 5 shows effect of α_d on the training and testing accuracy of OMLA. It can be observed from the figure that the accuracy of OMLA is acceptable for values of α_d in the interval $[0, 0.25]$. As α_d is increased further, the accuracy of OMLA goes down. This may be because OMLA discards more spike patterns from the learning process for high values of α_d. Based on this, a suitable range for setting α_d is [0, 0.25]. For all the experiments reported in this paper, the value of α_d is fixed at 0.25. In these experiments, we observed that α_d had a small effect on the number of neurons added by OMLA.

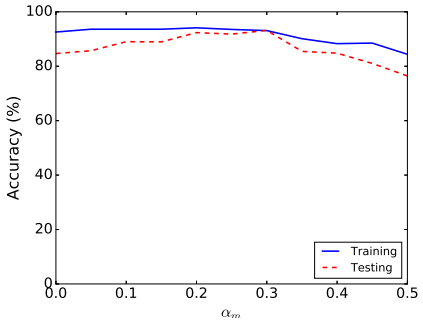


Figure 6: Effect of margin threshold (α_m) on the training and testing accuracy of OMLA

Effect of margin threshold (α_m): Figure 6 shows the training and testing accuracy of the OMLA for values of α_m in the interval $[0, 0.5]$. It can be observed from the figure that there is a small change in the training/testing accuracy of OMLA for values of α_m in the interval $[0, 0.3]$. In the same interval, the testing accuracy of OMLA varies by 6%. For values of α_m higher than 0.3, both training and testing accuracy of OMLA start decreasing. Based on this, an appropriate range for setting α_m is $[0, 0.3]$. For all the experiments conducted in this paper, the value of α_m is fixed at 0.3. In this range, α_m had a small effect on the number of neurons added by the OMLA.

5.2. Performance Comparison

In this section, the performance of OMLA is studied using five benchmark data sets from the UCI machine learning repository [20]. Also, its performance is compared with other existing online and batch learning algorithms for spiking neural networks. For online learning algorithms, a comparison was made with Online Spiking Neural Network (OSNN) [35] and the online version of the SRESN classifier [8]. With regards to the batch learning algorithm, a comparison was done with three well-known batch learning algorithms for SNNs, viz. SpikeProp [4], Synaptic Weight Association training (SWAT) [34] and the batch version of the SRESN classifier. The results for both the online version of SRESN (Online SRESN) classifier and the batch version of SRESN (Batch SRESN) classifier have been reproduced from [8]. Table 1 highlights the details of the data sets used for comparison. The table provides information about the number of features, number of classes and number of training/testing spike patterns for the data sets used in comparison. The Landsat data set was only used to evaluate the performance of the OMLA on a large data set. The performance of other algorithms has not been evaluated on Landsat data set due to the large computational requirements.

Table 1: Description of the data sets used for comparison

Data set	# Features	# Classes	# Spike Patterns	
			Training	Testing
Iris	4	3	75	75
Breast cancer	9	2	350	333
Liver	6	2	170	175
PIMA	9	2	384	384
Ionosphere	34	2	175	176
Landsat	36	6	4435	2000

For every data set, ten random sets are generated using the same number of training and testing spike patterns, as suggested in [2] (for Landsat, a single

fold was used to minimize computational effort required). The training and testing accuracy are computed for all the sets and the mean along with the standard deviation for the experiments is reported. The results for all the other algorithms have been generated by us except for the OSNN whose results have been reproduced from [35]. As in the original SpikeProp paper [4], the results for SpikeProp are generated using 16 delays per receptive field, a learning rate of 0.0075, coding interval of 4 milliseconds and a time constant of 7 milliseconds. The number of neurons in the hidden layer for SpikeProp is determined using the constructive-destructive procedure [31]. For SWAT, the important parameters are c_o and the maximum height of the plasticity window (A_p). The other parameters pertaining to the neuron model and frequency filtering are set as mentioned in the original paper [34]. It has been highlighted in the original paper that, a suitable value of c_o depends on the number of epochs required for convergence. We observed during the experiments that, SWAT converged within 500 epochs for all the data sets evaluated in this paper. Hence, c_o was set to 4000 as in the original paper. The impact of A_p is similar to the effect of learning rate on other learning algorithms and has been set in the interval $[0.1, 0.5]$ in the original paper. In this work, A_p is fixed at 0.1 to avoid oscillations in the learning process. For OMLA, the parameter values for delete

Table 2: Parameter values for novelty threshold (α_n) and learning rate (α_s) for benchmark data sets used for comparison

Data set	Novelty	Update
	Threshold (α_n)	Factor (α_s)
Iris	0.70	0.06
Breast cancer (BC)	0.96	0.06
Liver	0.98	0.05
PIMA	0.80	0.04
Ionosphere (ION)	0.73	0.09
Landsat	0.73	0.1

threshold and margin threshold in OMLA are fixed at 0.25 and 0.3, respectively. The parameter values for the novelty threshold (α_n) and the learning rate (α_s) for all the data sets, have been selected using 10-fold cross validation and are given in Table 2.

Table 3 shows the architecture and the results of the performance evaluation of the online learning algorithms. The architectures of OMLA and Online SRESN are shown in the format ($m : K$) as they employ two layered networks. The other algorithms employ a three layered architectures, hence, the number of hidden neurons is also shown. For evolving learning algorithms (OSNN, SRESN and OMLA) the architecture shows the range of neurons added by the learning algorithm for the ten random trials. It may be noted that the number of input neurons in the architecture of the different learning algorithms is equal to the product of the number of features and the number of receptive fields used for population coding.

Table 3: Performance comparison of OMLA with OSNN and the Online SRESN

Data set	Benchmark criterion	OMLA	OSNN	Online SRESN
Iris	Architecture	24:(5-7)	48:(7-21):3	24:(6-11)
	Training	97.9(0.7)	87.2(4.1)	92.7(4.2)
	Testing	97.9(0.7)	86.1(6.7)	93.0(5.7)
BC	Architecture	54:2	54:(10-16):2	54:(5-8)
	Training	97.4(0.4)	91.1(2.0)	93.9(1.8)
	Testing	97.8(0.4)	90.4(1.8)	94.0(2.6)
Liver	Architecture	36:(12-15)	36:(4-7):2	36:(5-8)
	Training	69.9(2.3)	58.7(2.2)	59.8(1.2)
	Testing	67.7(1.8)	56.7(1.8)	57.4(1.1)
PIMA	Architecture	54:20	54:(8-18):2	54:(6-12)
	Training	78.6(1.7)	68.2(2.0)	67.0(0.8)
	Testing	77.9(1.0)	63.5(3.0)	66.1(1.4)
ION	Architecture	204:(19-25)	204:(4-11):2	204:(6-13)
	Training	94.0(1.7)	76.7(2.4)	85.1(1.9)
	Testing	93.5(0.5)	76.6(4.8)	79.3(3.0)

It can be observed from Table 3 that, OMLA performs significantly better than the other online learning algorithms. For further discussion, the performance of OMLA is compared only with the performance of Online SRESN as Online SRESN performs better than OSNN. For simple problems like Iris flower classification and Wisconsin breast cancer, the performance of OMLA is 3% to 4% better than the performance of Online SRESN. For low dimensional problems with lower separability like Liver and PIMA, OMLA performs 10-11% better than Online SRESN. For a high dimensional problem like Ionosphere, OMLA performs 14% better than Online SRESN. Next, a statistical analysis of the performance comparison is presented.

Statistical analysis of performance comparison: A one-way ANOVA [13] test was conducted to analyze the results of the performance comparison between OMLA and other online learning algorithms. The statistical test was conducted with the null hypothesis that the performance of the three algorithms do not differ significantly. If the p-value for the computed F-statistic is lower than 0.05 (95% confidence interval) then the null hypothesis is rejected. In this study, the mean testing accuracy of the three algorithms for the five data sets represents three different groups and ANOVA monitors the variations between the groups. An F-statistic of 31.87 was obtained for group-wise variation which corresponds to a p-value of 0.0002. Hence, one can reject the null hypothesis with a 95% confidence interval. Thereafter, a pairwise comparison was performed between the three classifiers using the Bonferroni [11, 12] method. The observed p-values for the pairwise comparison of OMLA and Online SRESN was 0.0015 and for the pairwise comparison of OMLA and OSNN was 0.0002. Since, both the p-values are lower than 0.05 (95% confidence interval), it can be concluded that OMLA performs better than the other algorithms used for comparison with a 95% confidence interval. Next, the performance results of OMLA are compared with other existing batch learning algorithms for spiking neural networks.

Table 4 shows the results of comparison with batch learning algorithms. It can be observed from the table that the OMLA performs better than or

Table 4: Performance comparison of OMLA with SpikeProp, SWAT and the Batch SRESN

Data set	Benchmark criterion	OMLA	SpikeProp	SWAT	Batch SRESN
Iris	Architecture	24:(5-7)	25:10:3	24:312:3	24:(6-10)
	Training	97.9(0.7)	97.2(1.9)	96.7(1.4)	96.9(1.0)
	Testing	97.9(0.7)	96.7(1.6)	92.4(1.7)	97.3(1.3)
	# Epochs	1	1000	500	102
BC	Architecture	54:2	55:15:2	54:702:2	54:(8-12)
	Training	97.4(0.4)	97.3(0.6)	96.5(0.5)	97.7(0.6)
	Testing	97.8(0.4)	97.2(0.6)	95.8(1.0)	97.2(0.7)
	# Epochs	1	1000	500	306
Liver	Architecture	36:(12-15)	37:15:2	36:468:2	36:(6-9)
	Training	69.9(2.3)	71.5(5.2)	74.8(2.1)	60.4(1.7)
	Testing	67.7(1.8)	65.1(4.7)	60.9(3.2)	59.7(1.7)
	# Epochs	1	3000	500	715
PIMA	Architecture	54:20	55:20:2	54:702:2	54:(9-14)
	Training	78.6(1.7)	78.6(2.5)	77.0(2.1)	70.5(2.4)
	Testing	77.9(1.0)	76.2(1.8)	72.1(1.8)	69.9(2.1)
	# Epochs	1	3000	500	254
ION	Architecture	204:(19-25)	205:25:2	204:2652:2	204:(16-23)
	Training	94.0(1.7)	89.0(7.9)	86.5(6.7)	91.9(1.8)
	Testing	93.5(0.5)	86.5(7.2)	90.0(2.3)	88.6(1.6)
	# Epochs	1	3000	500	1018
Landsat	Architecture	216-30	101-25-6[1]	-	-
	Training	91.0	87(0.5)	-	-
	Testing	90	85.3(0.3)	-	-
	# Epochs	1	60000	-	-

[1] SpikeProp results have been reproduced from [4]. The Landsat data consists of 3x3 image patches with 4 channels that amounts to a total of 36 features. SpikeProp averages across the channels to obtain 4 features. It uses one bias neuron and 25 receptive fields per feature resulting in 101 input neurons.

similar to other batch learning algorithms but, it requires a single presentation of training spike patterns whereas other algorithms require multiple epochs for learning. For simple problems like Iris flower classification and Wisconsin breast cancer, the performance of all the algorithms is similar. For other problems,

further discussion is restricted to a comparison with SpikeProp and SWAT as SpikeProp performs better than the other batch learning algorithms for all the other problems, except in case of Ionosphere, where SWAT performs better than SpikeProp. For low dimensional problems with lower separability like Liver and PIMA, OMLA performs 1-2% better than SpikeProp and 5-7% better than SWAT. For a high dimensional problem like Ionosphere, OMLA performs 7% better than SpikeProp and 3% better than SWAT. For a data set with large number of samples like Landsat, OMLA performs 5% better than SpikeProp.

These observations clearly highlight that the utilization of global information present in the network, as well as the local information present in the input spike patterns, help the meta-neuron based learning rule in effectively updating the synaptic weights in one-shot.

6. Conclusions

In this paper, a spiking neural network architecture with a meta-neuron that envelopes the presynaptic and postsynaptic neurons has been presented. The concept of the meta-neuron is inspired by heterosynaptic nature of astrocytes in brain. It estimates a weight sensitivity modulation factor for the synapses in the network based on both, global information ('synaptic weights') and the local information ('input spike patterns'). The meta-neuron based learning rule adapts the synaptic weights based on the weight sensitivity modulation factor and the difference in the postsynaptic potential for precisely shifting the spike times of output neurons. Using this learning rule, an online meta-neuron based learning algorithm has been developed for an evolving spiking neural classifier. The performance of the OMLA is compared with both the other existing online and well-known batch learning algorithms for spiking neural networks using the benchmark pattern classification data sets from the UCI machine learning repository. In comparison to other online learning algorithms OMLA performs better with a 95% confidence level. With regards to other batch learning algorithms as well, the OMLA performs better using one-shot learning because it

utilizes both, the local and global information within the network.

The future work on the meta-neuron will focus on development of learning techniques for deep SNNs. Recent research [36, 27] on deep networks have proven their effectiveness in dealing with complex problems.

Acknowledgment

We thank the reviewers for their comments that helped in improving the overall quality of the publication. Also, we wish to extend our thanks to the Ministry of Education (MoE), Singapore, for providing financial support through tier I (No. M4011269) funding to conduct this study.

References

[1] Alfonso Araque, Vladimir Parpura, Rita P Sanzgiri, and Philip G Haydon. Tripartite synapse: glia, the unacknowledged partner. *Trends in neurosciences*, 22(5):208–215, 1999.

[2] GS Babu and S Suresh. Sequential projection-based metacognitive learning in a radial basis function network for classification problems. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2):194–206, 2013.

[3] Elie L Bienenstock, Leon N Cooper, and Paul W Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of neuroscience*, 2(1):32–48, 1982.

[4] Sander M. Bohte, Joost N. Kok, and Han La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[5] Eric A Bushong, Maryann E Martone, Ying Z Jones, and Mark H Ellisman. Protoplasmic astrocytes in CA1 stratum radiatum occupy separate anatomical domains. *The Journal of neuroscience*, 22(1):183–192, 2002.

[6] Ami Citri and Robert C Malenka. Synaptic plasticity: multiple forms, function, and mechanisms. *Neuropsychopharmacology*, 33(1):18–41, 2008.

[7] Yang Dan and Mu Ming Poo. Spike timing-dependent plasticity: from synapse to perception. *Physiological reviews*, 86(3):1033–1048, 2006.

[8] S Dora, K Subramanian, S Suresh, and N Sundararajan. Development of a self-regulating evolving spiking neural network for classification problem. *Neurocomputing*, 171:1216–1229, 2016.

[9] Shirin Dora, Savitha Ramasamy, and Sundaram Suresh. A basis coupled evolving spiking neural network with afferent input neurons. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2013.

[10] Shirin Dora, Sundaram Suresh, and Narsimhan Sundararajan. A sequential learning algorithm for a spiking neural classifier. *Applied Soft Computing*, 36:255–268, 2015.

[11] Olive Jean Dunn. Estimation of the medians for dependent variables. *The Annals of Mathematical Statistics*, 30(1):192–197, 1959.

[12] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[13] R A Fisher. On the probable error of a coefficient of correlation deduced from a small sample. *Metron*, 1(4):3–32, 1921.

[14] Răzvan V. Florian. The chronotron: A neuron that learns to fire temporally precise spike patterns. *PLoS ONE*, 7(8):1–27, 2012.

[15] Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single neuron, Populations and Plasticity*. Cambridge University Press, 2002.

[16] Jesse Goldberg, Knut Holthoff, and Rafael Yuste. A problem with hebb and local spikes. *Trends in neurosciences*, 25(9):433–435, 2002.

[17] Michael M Halassa, Tomasso Fellin, Hajime Takano, Jing-Hui Dong, and Philip G Haydon. Synaptic islands defined by the territory of a single astrocyte. *The Journal of neuroscience*, 27(24):6473–6477, 2007.

[18] Nikola Kasabov and Elisa Capecci. Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes. *Information Sciences*, 294:565–575, 2015.

[19] Nikola Kasabov, Kshitij Dhoble, Nuttapod Nuntalid, and Giacomo Indiveri. Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, 41:188–201, 2013.

[20] M. Lichman. UCI Machine Learning Repository, 2013.

[21] Wolfgang Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9(2):279–304, 1997.

[22] Wolfgang Maass. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. *Institute for Theoretical Computer Science, Technische Universitaet Graz, Austria, Technical Report*, 1999. URL `http://www.igi.tugraz.at/psfiles/90.pdf`.

[23] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297):213–215, 1997.

[24] Gertrudis Perea, Marta Navarrete, and Alfonso Araque. Tripartite synapse: astrocytes process and control synaptic information. *Trends in neurosciences*, 32(8):421–431, 2009.

[25] Filip Ponulak and Andrzej Kasiński. Supervised learning in spiking neural networks with resume: Sequence learning, classification and spike shifting. *Neural Computation*, 22(2):467–510, 2010.

[26] José Principe, Neil R. Euliano, and W. Curt Lefebvre. *Neural and Adaptive Systems: Fundamentals through Simulations*. Wiley, 1999.

[27] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, pages 525–542, 2016.

[28] Rufin Van Rullen and Simon J Thorpe. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13(6):1255–1283, 2001.

[29] Tao Song, Pan Zheng, ML Dennis Wong, and Xun Wang. Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control. *Information Sciences*, 372(380-391), 2016.

[30] Ioana Sporea and André Grüning. Supervised learning in multilayer spiking neural networks. *Neural Computation*, 25(2):473–509, 2013.

[31] S Suresh, SN Omkar, V Mani, and TN Guru Prakash. Lift coefficient prediction at high angle of attack using recurrent neural network. *Aerospace Science and Technology*, 7(8):595–602, 2003.

[32] Sundaram Suresh, Keming Dong, and HJ Kim. A sequential learning algorithm for self-adaptive resource allocation network classifier. *Neurocomputing*, 73(16):3012–3019, 2010.

[33] Simon Thorpe and Jacques Gautrais. Rank order coding. In *Computational Neuroscience*, pages 113–118. Springer, 1998.

[34] John J. Wade, Liam J. McDaid, Jose A. Santos, and Heather M. Sayers. SWAT: A Spiking Neural Network Training Algorithm for Classification Problems. *IEEE Transactions on Neural Networks*, 21(11):1817–1830, 2010.

[35] Jinling Wang, Ammar Belatreche, Liam Maguire, and Thomas Martin McGinnity. An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing*, 144:526–536, 2014.

[36] Yunhe Wang, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. CNNpack: Packing Convolutional Neural Networks in the Frequency Domain. In *Advances In Neural Information Processing Systems*, pages 253–261, 2016.

[37] Simei Gomes Wysoski, Lubica Benuskova, and Nikola Kasabov. Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, 23(7):819–835, 2010.